Amendment to the Specification

On page 1, please amend the title to read as follows:

METHOD FOR DYNAMIC MANAGEMENT OF TCP REASSEMBLY BUFFERS

On page 1, please amend paragraph [0001] as follows:

[0001] The present invention is related to data transfer. More particularly, the present invention provides a method ~~and system~~ for dynamic management of Transmission Control Protocol (TCP) reassembly buffers in hardware (e.g., in a TCP/IP offload engine (TOE)).

On pages 2-4, please delete paragraphs [0008] and [0009], and amend paragraphs [0005]-[0007] and [0010] as follows:

[0005] The present invention provides a method ~~and system~~ for the dynamic management of TCP reassembly buffers in hardware. Dynamic memory management significantly improves the flexibility and scalability of available memory resources. The major challenge of dynamic memory management of TCP reassembly buffers in hardware, however, is to reduce its associated performance penalty and to bring its performance as close as possible to the performance achieved by static memory management methods. The present invention accomplishes these goals.

[0006] The present invention provides a method ~~and system~~ for flexible dynamic memory management of TCP reassembly buffers, allowing efficient hardware implementation. In addition, the method ~~and system~~ of the present invention allow combined dynamic and static memory management using the same hardware implementation. The decision to use dynamic or

O.K. to enter
RA 1/7/09

Amendments to the Abstract

On page 22, Abstract of the Disclosure, please amend the paragraph as follows:

A method ~~and system~~ for dynamic management of Transmission Control Protocol (TCP) reassembly buffers in hardware (e.g., in a TCP/IP offload engine (TOE)). The method comprises: providing a plurality of data blocks and an indirect list; pointing, via entries in the indirect list, to allocated data blocks in the plurality of data blocks that currently store incoming data; if a free data block in the plurality of data blocks is required for the storage of incoming data, allocating the free data block for storing incoming data; and, if an allocated data block in the plurality of data blocks is no longer needed for storing incoming data, deallocating the allocated data block such that the deallocated data block becomes a free data block.